

# Travaux dirigés micro 8 bits, GE3

David Delfieu

January 16, 2026

## 1 *PORTS* d'entrées-sorties

On dispose d'un d'un  $\mu C$  de type *ATMEGA8* et l'on souhaite avec ses *PORTS* d'E/S réaliser les connections suivantes :

- 3 leds sont connectées sur les voies  $PB_7, PB_6, PB_5$  via des résistances à la masse.
- 5 boutons de commande sont connectés sur les voies  $PB_4, PB_3, PB_2, PB_1, PB_0$ .
- 8 autres led sont connectées sur  $PORT_C$  connnectéers de la même manière.

On donne le cahier des charges suivant : À l'appui d'un seul des 5 boutons de commande, un chenillard démarre en partant de  $PC_0$  jusqu'à  $PC_5$ . La led connectée à la  $PC_0$  va s'allumer d'abord pendant 500 ms puis celle de  $PC_1, \dots$  jusqu'à  $PC_5$ , après  $PC_5$ , c'est la led connectée sur  $PC_0$  qui s'allume et le cycle repart. Ce chenillard se déplace avec une fréquence de 2 Hz pendant 10 s.

Dans le même temps on affichera sur  $PB_7, PB_6, PB_5$  le numéro du bouton de commande qui aura été activé. Par exemple, si le bouton 3 a été activé on aura sur ces bits la valeur 011.

### 1.1 Algorithme

Donner l'algorithme de premier niveau du code à réaliser.

### 1.2 Initialisation

On s'intéresse dans un premier temps à l'initialisation des *PORTS* :

1. Coder en décimal, hexadécimal et binaire la configuration des *PORTS* *B* et *C*
2. On souhaite améliorer la sûreté de fonctionnement en activant les résistances de tirages de pull-down sur les boutons de commande. Donner l'intérêt des résistances de tirage, puis le code pour les activer.

### 1.3 Affichage

On cherche ensuite à gérer les différents affichages. Pour simplifier l'écriture on définit les masques suivant  $M_1 = 0x1F$  et  $M_2 = \overline{M_1} = 0xE0$ . Considérons une variable de type char (8 bits) *X* initialisée à la valeur 177.

1. Donner le résultat du calcul binaire  $X \& M1$  et  $X | M2$
2. A l'aide de l'annexe, trouver une écriture alternative à ces masques.
3. Donner le code complet correspondant à l'algorithme.

## 1.4 Lecture de Port

Le banc de leds est maintenant connecté à  $\text{PORT}_C$ .  $\text{PORT}_B$  est utilisé en entrée. Considérons le programme suivant :

```
#define ALLUME 0xFF
#define ETEINT 0x00
#define TOUTES 0xFF
#define AUCUNE 0x00
#define PREMIER 0b00000001

int main(void){
    int test, i;

    DDRB = AUCUNE;
    SFIOR &= ~(1<<PUD);
    PORTB = PREMIER; //Resistance de tirage sur PB0

    DDRC = TOUTES;
    PORTC = ETEINT;

    while(1){
        test = PINB & PREMIER;
        if (test == 0) PORTC = ALLUME;
        else PORTC = ETEINT;
        for (i = 0; i < 500; i++) _delay_ms(10); //on attend 5 secondes
    }
}
```

- Que fait ce programme ?
- Comment définiriez-vous le temps de cycle de ce programme ?
- Si l'entrée  $PB_0$  passe à 0, le  $\text{PORT}_C$  est-il immédiatement commuté ?
- Quel élément du  $\mu C$  pourriez-vous utiliser pour obtenir l'exécution d'une tâche **aussitôt que**  $PB_0$  passe à zéro ?

## Annexe

- Mise à un par masque avec opérateur “ou” |:
- Mise à zéro par masque avec opérateur “et” &:
- Inversion avec opérateur “xor” ^:

```
// Initialisation ou modification ?
// Utiliser |= ou &= quant on veux modifier partiellement un registre
// Pour initialiser simplement un registre on utilise l'affectation =
PORTD = 0b0000 0011;
PORTD |= 1<<PORTD4; // PORTD = Ob0001 0011;
PORTD = 0b0000 0011;
PORTD = 1<<PORTD4; // PORTD = Ob0001 0000;

// SET 4 bits de poids faible d'un port SANS MODIFIER les autres bits
```

```

PORTB = PORTB | 0x0F; // ou bien
PORTB |= 0xF0;

// RAZ des 4 bits de poids faible SANS MODIFIER les autres bits
PORTB=PORTB & 0xF0; // ou bien
PORTB &= 0xF0;

// RAZ du bit 4 du PORT D SANS MODIFIER les autres bits
PORTD &= ~(1<<PORTD4);

// Commutation du bit 4 du PORT D SANS MODIFIER les autres bits
PORTD ^= (1<<PORTD4);

```

## 2 Interruptions externes

L'ATMEGA8 a deux pattes d'interruption externe  $INT_0$  et  $INT_1$ . Toutes les pattes de l'ATMEGA8 (sauf alimentation) ont plusieurs fonctions. Ainsi, la patte  $PD_2$  correspond à l'interruption externe  $INT_0$  comme la patte  $PD_3$  pour l'interruption  $INT_1$ . Afin d'utiliser ces interruptions externes, on peut distinguer trois phases de programmation :

1. Donner les instructions de masque sur les registres  $SREG$ ,  $GICR$  et  $MCUCR$  qui permettent de configurer les bits qui autorisent le déclenchement d'une interruption lorsqu'un front montant intervient sur la patte  $INT_0$ .
2. Compléter le programme présenté en section 1.4 de manière à ce que  $PORT_C$  soit commutée à chaque front montant sur la patte  $INT_0$ .

$MCUCR : SE\ SM_2\ SM_1\ SM_0\ ISC_{11}\ ISC_{10}\ ISC_{01}\ ISC_{00}$

$GICR : INT_1\ INT_0\ ---\ IVSEL\ IVCE$

$SREG : IT\ H\ S\ V\ N\ Z\ C$

Table 1: ISC11 et ISC10

ISC11	ISC10	Trigger
0	0	Niveau bas (Tant Que)
0	1	Front montant ou front descendant
1	0	Front descendant
1	1	Front montant

Table 2: ISC01 et ISC00

ISC01	ISC00	Trigger
0	0	Niveau bas (Tant Que)
0	1	Front montant ou front descendant
1	0	Front descendant
1	1	Front montant

## 3 Chenillard et *Interrupt Externe*

Faire un chenillard par défaut qui tournera sur  $PORT_B$  dans un sens que l'on appellera sens positif.

- Pour un front montant sur  $PD_3$  faire tourner dans le sens négatif.
- Pour un front descendant sur  $PD_2$  faire tourner dans le sens positif.

## 4 Chenillard et *Convertisseur Analogique*

Faire un chenillard sur  $\text{PORT}_B$ . Mettre en place une conversion analogique du bit  $PC_3$  qui réglera la vitesse du chenillard.

- On fonctionnera en mode lecture analogique scrutative.
- Puis on fonctionnera en mode free-running.

## 5 Leds et *Timer 2*

Faire clignoter  $\text{PORT}_C$  avec une temporisation réalisée grâce à *Timer 2*. L'interruption de débordement du timer doit provoquer le clignotement.

## 6 *Timer 1 et Convertisseur Analogique*

- A l'aide du timer 1, générer une *PWM* centrée avec un rapport cyclique constant de 60% sur  $PB_2$  et une fréquence de hachage de 12 khz. Observer les signaux à l'oscilloscope.
- Faire varier la fréquence avec une conversion analogique sur  $PC_2$  selon un mode échantillonné : L'interruption de débordement du timer 1 doit lancer la conversion.